

Homework 3

Adversarial Search

Due: 6:00pm on 10/07/09

Problem 3.1

Consider the labeling of 2 player game trees not with a single value, but with a pair of values: e.g., $(x, -x)$. The first value denotes the *payoffs* to player 1 and the second value denotes the payoffs to player 2. Extend this labeling and the MINIMAX algorithm to handle N player games.

In games of $N \neq 2$ players, it no longer makes sense to view the objectives of the players as maximizing and minimizing, particularly in non-zero sum games. One conceivable node labeling uses tuples $(x_1, \dots, x_N) \in \mathbb{R}^N$ without assuming $\sum_{i=1}^N x_i = 0$. The MINIMAX algorithm extends naturally to the case of $N \geq 1$ player, non-zero sum games. The following N -player versions of MINIMAX, called MAXIMAX, assume all players are maximizing.

BACKUP1MAX(G, n, m)
Inputs game G , node n , parent m
Output updated game tree values
a. let $i = l(m)$
b. $v_i(m) = \max\{v_i(m), v_i(n)\}$

Problem 3.2

Given a finite set S and a binary operation $* : S \times S \rightarrow S$, the tuple $(S, *)$ is called a group if and only if the following properties hold:

- Neutral Element: $\exists n \in S \forall a \in S : n * a = a = a * n$
- Associative: $\forall a, b, c \in S : (a * b) * c = a * (b * c)$
- Inverse: $\forall a \in S : \exists b \in S : a * b = n = b * a$

Prove the following:

- a. The neutral element is unique.
- b. $\forall a \in S$, a 's inverse element is unique.

Solution:

<p>MAXIMAX-DFS(G, x)</p> <p>Inputs game G, node x</p> <p>Output value of node $v(x)$</p> <p>Initialize $O = \{x\}$</p>
<p>while (O is not empty) do</p> <p> a. choose <i>first</i> node $n \in O$</p> <p> b. if $n \in T$, evaluate $v(n)$</p> <p> c. if $v(n) \neq \perp$</p> <p> (a) if $n \neq x$</p> <p> i. $m = \delta^{-1}(n)$</p> <p> ii. BACKUP1MAX(G, n, m)</p> <p> (b) <i>delete</i> n from O</p> <p> d. else if $v(n) = \perp$</p> <p> (a) $v(n) = -\infty$</p> <p> (b) prepend $\delta(n)$ to <i>front</i> of O</p> <p>return $v(x)$</p>

Table 1: Maximax Depth-First Search.

Suppose $\exists w \in S$ s.t. $w * a = a = a * w \forall a \in S$

then $w = w * n = n$

so $w = n$ and n is unique.

Suppose $a * b = n = b * a$ and $a * c = n = c * a$ then

$b = b * n = b * (a * c) = (b * a) * c = n * c = c$ so $b = c$ and the inverse is unique.

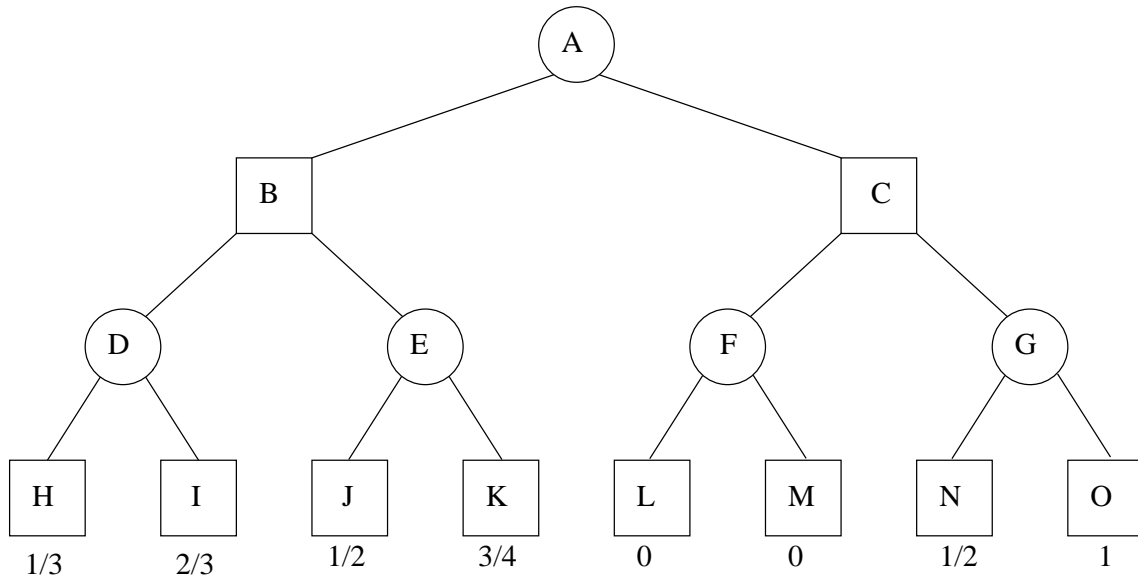
Problem 3.3

Run **Alpha-Beta-Search** (page 170 in Russel and Norvig) on the game tree given in figure. Numbers below a node are the result of calling **Utility** on that node.

- Fill in the table with the values of α and β when **Max-Value** or **Min-Value** is called on each node.
- Put a slash through any branch that is pruned: *e.g.*, if the search visits node X but does not visit its child node Y , put a slash through the edge connecting X to Y .

<p>MAXIMAX(G, x)</p> <p>Inputs game G, root node x of subgame</p> <p>Output maximax value of subgame rooted at x</p> <p>Initialize $\alpha = -\infty$</p>
<p>a. if $x \in T$, return $v(x)$</p> <p>b. else for all $y \in \delta(x)$</p> <p style="padding-left: 40px;">(a) $\alpha = \max\{\alpha, \text{MAXIMAX}(G, y)\}$</p> <p>c. return α</p>

Table 2: Recursive Maximax.



Node	α	β
A		
B		
C		
D		
E		
F		
G		
H		
I		
J		
K		
L		
M		
N		
O		

This solution uses the recursive $\alpha\beta$ Pruning algorithm given on page 12 of Amy's notes on Adversarial Search:

Node	α	β
A	$-\infty$	∞
B	$-\infty$	∞
C	$2/3$	∞
D	$-\infty$	∞
E	$-\infty$	$2/3$
F	$2/3$	∞
G	—	—
H	$-\infty$	∞
I	$1/3$	∞
J	$-\infty$	$2/3$
K	$1/2$	$2/3$
L	$2/3$	∞
M	$2/3$	∞
N	—	—
O	—	—

—: This node is pruned.

Problem 3.4

Suppose you are given a two-player, zero-sum game $G = \langle \{\text{Max}, \text{Min}\}, X, S, T, \delta, l, v \rangle$. To find the minimax value of the game, while expanding as few nodes as possible, you would probably use $\alpha\beta$ Pruning over Minimax.

Now suppose you also have a good, but not perfect, static evaluation function $e : X \rightarrow \mathbb{R}$ (a static evaluation function takes a game state and returns a guess based on that state, returns a number that is a good guess as to the value of that state). Design an algorithm that is expected to perform better (expand fewer nodes).

Specifically:

- Your algorithm should return the exact minimax value, not an approximation.
- You may assume that S , the set of start states, is a singleton, with distinct element s .
- You are not given anything beyond G and e . If you design an algorithm, it must work given only those two inputs.
- You may assume that applying the static evaluation function to a node does not use excessive time or space.
- Speed is not the primary concern (the number of visited nodes is), but algorithms that are needlessly slow will be penalized.
- An algorithm does not need to visit strictly fewer nodes in all cases to be considered an improvement: *e.g.*, $\alpha\beta$ Pruning does not visit strictly fewer nodes than MiniMax in all cases, but it is still considered better than Minimax because it never visits more nodes and it sometimes visits strictly fewer nodes.

You may present your answer in pseudo-code or you may write a clear explanation. You should also provide a brief explanation of why your algorithm performs better than $\alpha\beta$ Pruning.

Solution: You can use e to sort the nodes before each recursive call of $\alpha\beta$ Pruning. At Maxie nodes one would order them from largest to smallest and at Minnie from smallest to largest. Initially, the most promising path would be explored. Even if this path does lead to a minimax solution, hopefully many of the less promising children could be pruned.

Problem 3.5

Suppose the CS141 staff (Yuri, Ilke, Meinolf, and Matt) all live in houses on the same side of the street, numbered, left to right, 12, 14, 16, and 18. Solve the following constraint program and show your reasoning.

- a. Meinolf drives a Kia.
- b. Ilke has a monkey.
- c. Matt eats ostrich.
- d. Yuri lives in the last house.
- e. The Plymouth driver lives in the first house.
- f. The BMW driver lives to the left of the buffalo eater.
- g. The wine drinker and the beer drinker are neighbors.
- h. The sake drinker has a neighbor with a pet llama.
- i. The dog owner and the wine drinker are not neighbors.
- j. The fish owner has a neighbor who eats deer.
- k. The dog owner drinks ouzo.
- l. The BMW driver eats duck.
- m. The Peugeot driver lives to the right of the sake drinker.
- n. The ouzo drinker eats buffalo.
- o. The deer eater keeps a llama as a pet.
- p. Everyone keeps a different pet.
- q. Everyone enjoys a different drink.
- r. Everyone drives a different make of car.
- s. Everyone eats a different type of meat.
- t. Everyone lives in a different house.

Who drinks the sake?

Solution:

Ilke drinks the sake!

We expected clear reasoning, or a table to explain why.